**Proceedings of DETC 03**
**ASME 2003 Design Engineering Technical Conferences**
**and Computers and Information in Engineering Conferences**
**Chicago, Illinois, September 2 – 6, 2003**

# DETC2003/CIE-48239

# ENHANCING VIRTUAL PRODUCT REPRESENTATIONS FOR ADVANCED DESIGN REPOSITORY SYSTEMS

**Matt R. Bohm**
**and Robert B. Stone, Ph.D.**
Design Engineering Laboratory
Department of Basic Engineering
University of Missouri – Rolla
Rolla, Missouri 65401-0210

**Simon Szykman, Ph.D.**
Manufacturing Systems Integration Division
National Institute of Standards and Technology
100 Bureau Drive, Stop 8263
Gaithersburg, MD 20899-8263

## ABSTRACT

This paper describes the transformation of an existing set of heterogeneous product knowledge into a coherent design repository that supports product information archival, storage and reuse. Existing product information was analyzed and compared against desired outputs to ascertain what information management structure was needed to produce design resources pertinent to the design process. Several test products were cataloged to determine what information was essential without being redundant in representation. This set allowed for the creation of a novel single application point of entry for product information that maintains data consistency and allows information be easily exported. The exported information takes on many forms that are valuable to the design process such as a bill of materials and component function matrix. Enabling technologies include commercial software, XML (eXtensible Markup Language) data, XSL (eXtensible Stylesheet Language) transformation sheets and HTML (HyperText Markup Language). Through this process researchers at the University of Missouri – Rolla (UMR) have been able to dramatically improve the way in which artifact data is gathered, recorded and used.

## 1 INTRODUCTION

As products become more complex there is an increased need for the designer or team of designers to be able to have access to a breadth of design information spanning a variety of disciplines. Consideration of many types of artifacts is necessary when searching for component solutions, in order to ensure a high quality product that meets the needs of the customer. Well suited to meet this need are design repositories – knowledge bases of heterogeneous product design knowledge that can be searched and reused. Design repositories support these types of activities in original design and redesign cases.

Over the course of several years of research and integrated design coursework at the University of Missouri-Rolla (UMR), a body of product design knowledge was developed for approximately fifty consumer products. This knowledge base, which included descriptive product information such as functionality, bills of materials and design structure matrices, lacked a standard interface, data consistency and the ability to output design representations with ease. Observing user interactions with the knowledge base, including design modeling activities as well as retrieval and reuse/redesign activities, revealed that these drawbacks served as a barrier to effective use of the knowledge base. By unifying these disparate components into a design repository, it has been possible to improve the utility of the knowledge base for viewing, searching and reusing the wealth of preexisting design knowledge.

This paper reports on research efforts conducted at UMR 1) to accurately identify the types of design knowledge required to support designer activities; 2) to represent information from this product knowledge base in a design repository system; and 3) to reuse that design knowledge for future product design. By following a National Institute of Standards and Technology (NIST) research effort in the area of design repositories, and by reviewing a design knowledge base system previously developed at UMR, a transformation has taken place in the way research activities at UMR catalog, view, and export design information. A key goal of the NIST design repository effort has been to generate a set of platform-independent data models that can easily be transferred from one system to another. NIST has proposed a set of information models that provide a generic, neutral format for capturing, storing and reusing product representation knowledge. Mappings of these information models into XML have been developed to facilitate system implementation and data exchange. Through the use of documentation and reports provided by NIST, as well as personal communications with NIST staff, UMR focused on developing the capability to export XML data in conformance with the NIST design repository representation class structure.

In addition to adopting the NIST-developed representation schemata, UMR researchers also had the goal of creating an application that could output design aids such as detailed bills of materials, matrices to support design computations and, ultimately, graphical functional models. A single, simplified point of entry for product information that integrate s well with product dissection processes (Otto & Wood 2001) was also desired. With these objectives identified, this paper examines the design knowledge base that had been previously developed at UMR, looking at how product information is collected and what information is necessary. The key pieces of design information gathered are functional models and a detailed bill of materials. From combinations of these two pieces of information, all other desired product representations can be derived. These two sets of data form the basis for transforming an emerging knowledge-based system into a more mature design repository-based tool.

## 2 BACKGROUND

In order to create a quality design repository, there are several elements of previous research that must be used to represent a product with consistency. This section starts out by reviewing commercially available software applications that resemble a design repository system, though do not fully fulfill the role of repositories in design. Next, section 2.2 focuses on product functionality, a key component of design information for product categorization, search and reuse. In particular, the functional basis is described and is presented as a means to describe product functionality.

### 2.1 COMMERCIAL DESIGN REPOSITORY SYSTEMS

There is currently no product on the market that is truly a design repository; however, there are several packages that contain elements of a design repository. Such computerized design packages can be grouped into three basic categories: 1) mechanical computer-aided design (MCAD) packages that augment traditional CAD models with more abstract design knowledge; 2) systems engineering toolsets which contain higher level design information that may be used to generate CAD models; and 3) systems modeling and simulation packages that have little or no interaction with traditional CAD packages. For the MCAD packages, the typical approach is to add layers of abstract design knowledge to the existing CAD model. For all categories, no standard language has evolved, though there is widespread use of the process of functional decomposition. Within such decompositions, whether for function or architecture, no standard exists concerning levels of abstraction. Finally, each package appears to use its own proprietary data structure to store the additional design knowledge. One representative example commercial package for each of the three categories is described for illustrative purposes.[1]

Unigraphics – UG/WAVE (www.ug.eds.com/ug/). UG/WAVE is a MCAD package. It adds abstract product design knowledge capability to the core Unigraphics CAD (or solid modeling) package. Product architecture information in a parametric product layout is captured in a "control structure." This appears to be similar to a functional modeling approach to product design, but is more form oriented. The module allows "what-if" evaluation of simplified design alternatives, making the necessary modifications to the rest of the design as necessary. It does store subsystem design knowledge such that it can be re-used in future products.

3SL – Cradle (www.threesl.com/). Cradle is a British systems engineering toolset composed of six modules that can be used together or separately. Its systems modeling component offers robust support of several modeling notations, including functional block diagrams, behavior diagrams and object oriented support. Design knowledge is stored in a single repository structure accessible by all modules. Cradle also allows high levels of re-use among subassemblies stored in its repository. While it is not a CAD package per se, it can export information to a variety of CAD formats.

Nu Thena Systems – Foresight (www.nuthena.com). Foresight is strictly a systems modeling and simulation tool. It takes a hierarchical approach to functional and architecture modeling, allowing as many levels of abstraction as desired. In addition to the functional and architecture model, a mapping between the two is stored as design knowledge. This provides a strong link between function and form design. The package does not interact with other CAD systems and the functional language used favors electronic systems.

The concept of a design repository is extremely useful in the context of automated design storage and retrieval packages. Although design repository-based systems do not exist in commercial form today, the review of current commercial offerings indicates that elements of the design repository concept are being adapted by mainstream commercial product development systems, and that industry is moving towards the vision of design repositories. No clear direction exists for its development, though. A standard repository structure, supported by fundamental functional and architecture modeling research, is needed to guide work in this area.

### 2.2 PRODUCT FUNCTIONALITY

Addressing the need for a clear vocabulary to describe product function, the functional basis has emerged as a standardized design language (Hirtz et al., 2002). It was formulated in concert with NIST to unify two similar, independent research efforts (Szykman et al., 1999; Stone and Wood, 2000). The functional basis consists of two sets of terminology: one containing action verbs to describe function, and a second containing nouns to describe flow. The functional basis spans all engineering domains while retaining independence of terms. The function set of the basis is broken down into eight categories termed the primary classes. These classes have further divisions, called the secondary and tertiary levels, that offer increasing degrees of specialization. The primary class represents the broadest definition of distinct function while the tertiary class provides a very specific description of function. The secondary level of the function set, containing twenty-one action verbs, is the most often used class of the basis. The primary class and secondary function terms are shown in Table 1.

The flow set of the functional basis allows for the associated

[1] Use of any commercial product or company names in this paper is intended to provide readers with information regarding the implementation of the research described, and does not imply recommendation or endorsement by the authors or their institutions.

**Table 1 - Function classes and their basic categorizations**

| | Branch | Channel | Connect | Control Magnitude | Convert | Provision | Signal | Support |
|---|---|---|---|---|---|---|---|---|
| Secondary | Separate | Import | Couple | Actuate | Convert | Store | Sense | Stabilize |
| | Distribute | Export | Mix | Regulate | | Supply | Indicate | Secure |
| | | Transfer | | Change | | | Process | Position |
| | | Guide | | Stop | | | | |

**Table 2 - Flow classes and their basic categorizations**

| Primary | Material | Signal | Energy | | |
|---|---|---|---|---|---|
| Secondary | Human | Status | Human | Electrical | Mechanical |
| | Gas | Signal | Acoustic | Electromagnetic | Pneumatic |
| | Liquid | | Biological | Hydraulic | Radioactive |
| | Solid | | Chemical | Magnetic | Thermal |
| | Plasma | | | | |
| | Mixture | | | | |

function's input and output flows to also be described. Similar to the function set, there are three distinct classes within the flow set of the functional language. Within the primary class of the flow set, there are three main categories used to describe flow: material, signal and energy – as popularized by Pahl and Beitz (1996). Each of these categories has the capability to represent the input or output of a function. The secondary class of this set has 20 nouns that are used to describe the type of flow. It is the secondary class of this basis that is primarily used when describing a product. The primary class and secondary flow terms are shown in Table 2. The tertiary level is omitted from Tables 1 and 2 for reasons of brevity, and can be found in (Hirtz et. al, 2002).

Using the functional basis to represent product functionality within the design repository allows product knowledge to be searched and categorized by their function. This abstraction allows the designer to focus on overall functionality and to develop more creative solutions for solving a design problem (McAdams and Wood, 2000).

## 3 ASSEMBLING PRODUCT DESIGN KNOWLEDGE COMPONENTS

In addition to product function, researchers at UMR have identified design information typically recorded and used in original and redesign settings which is not formally captured by current computational design tools. While product function is a core component of design representation, other higher-level descriptions (such as customer needs) and lower level representations (such as component dimensions) are needed to completely describe and, thus, archive product knowledge. The different representations chosen for this work are based on product information flow schemes (Shooter et al., 2000) and dissection processes (Sheppard, 1992; Otto and Wood, 2001) as well as information needed in a variety of modern design methods and tools (Pimmler and Eppinger, 1994; Campbell et al., 2000; McAdams and Wood, 2000; Wood and Verma, 2000; Campbell et al., 2001; Strawbridge et al., 2002; Wood et al., 2002; Stock et al., 2003). Eight types of design models that have been used to represent design information in support of the conceptual design process at UMR (and more generally as well) are identified below. Finally, Sections 3.2 and 3.3 describe the NIST-inspired design repository representation and outlines how it is designed to handle this new data.

### 3.1 INFORMATION MODELING AND CAPTURE IN THE UMR KNOWLEDGE-BASED SYSTEM

*Customer needs*: To record product information, existing products are utilized and operated, and a basic list of customer needs associated with the product is gathered. This is vital in identifying the product's purpose and market niche. It is important to understand the market area and/or level of performance that an individual product is expected to meet. To gather customer needs the product is operated by a designer and a survey is produced. The survey is geared to determine what features customers enjoy, how they feel about the product and if there is an overall purpose for the product. The questionnaire is also designed to find out what features or additions could be incorporated into the product and to establish a performance rating of the included features. The surveys are conducted with potential product customers while they operate the product. Customers are asked to rate areas on the survey on a 1-5 (5 being best and 1 being minimal) scale. Once a base of customers had been surveyed, the responses are averaged to determine the overall customer need weight (Hauser and Clausing, 1988; Otto, 1996; Urban and Hauser, 1993).

*Bill of Materials (BOM)*: A bill of materials is a detailed description of all of the artifacts within a given product. This provides an easy way for designers to see a simple breakdown of parts contained within a given product. Although artifact function descriptions are not traditionally used as part of BOM representations, in the context of research activities at UMR, such descriptions—represented as function and flow pairs—are associated with each artifact. For example, the artifact motor has the functional description of "convert electrical energy to mechanical energy." Additional information about an artifact's mass, dimensions, manufacturing processes or material composition is also recorded. The BOM is usually represented in a tabular format with the artifacts or part number listed on the left most column, where each row represents a different artifact. A partial BOM for an electric wok is shown in Table 3.

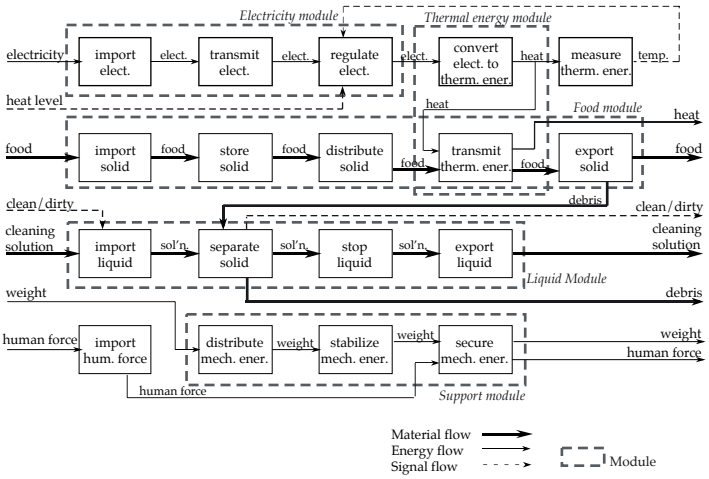
*Functional Model*: A functional model is a description of a product or process in terms of the elementary functions that are required to achieve its overall function or purpose. A graphical form of a functional model is represented by a collection of sub-

**Table 3 – A partial BOM for an electric wok**

| Part # | Component Number | Qty | Part Description | Part Color | Function (Sub-fct. Decsription) | Physical Parameters | Mfg. Process |
|---|---|---|---|---|---|---|---|
| A1 | 1 | 1 | Lid screw | Silver | Secure M.E. | Dia=0.3 " | Rolled |
| | 2 | 1 | Lid handle | Black | Stop Th.E. | Dia=2 " | Sawing |
| | 3 | 1 | Lid | Red | Import solid and export solid | Dome shaped with dia=12.25 " | Casting |
| A2 | 1 | 1 | Temp switch | Red | Regulate E.E. | Length=4 " | Processed casting |
| | 2 | 1 | Potentiometer mechanism | Silver/gold | Regulate E.E. | Length=2 " | Injection Molding |
| | 3 | 1 | Plug and cord | Black | Transfer and import E.E. | Two pronged plug, copper wires insulated with plastic | Drawing |
| | 4 | 1 | Socket | Silver | Transfer and import E.E. | Two pronged socket | Stamping |
| | 5 | 5 | Wire snaps | Silver | Transfer E.E. | | Stamping |
| | 6 | 1 | Heating coils | Silver | Convert EE to Thermal Energy, Sense thermal energy | Coil dia=0.5 " | Casting |
| A3 | 1 | 1 | Metal plate | Silver | Secure M.E. | Length=4 ", width=4 " | Casting |
| | 2 | 6 | Nuts to hold base | Silver | Secure M.E. | Dia=0.3 " | Rolled |
| | 3 | 1 | Metal base cover | Silver | Secure M.E. | Dia=6.5 " | Casting |
| | 4 | 2 | Nuts to hold metal base cover | Silver | Secure M.E. | Dia=0.3 " | Rolled |
| | 5 | 1 | Plastic support stand | Black | Stabilize M.E. | 4 legged stand length of length=5 " | Injection Molding |
| | 6 | 2 | Threaded base metals | Silver | Secure M.E. | Length=2.25 " | Casting |
| A4 | 1 | 2 | Handle screws | Black | Secure M.E. | Dia=0.4 " | Rolled |
| | 2 | 2 | Handles | Black | Import Hand,Stop Th.E. and distribute M.E. | Dim=3.5 " X 1.25 " | Injection Molding |
| | 3 | 1 | Pan | Red | Store solid, transfer Th.E., distribute M.E, distribute solid. | Dia=12.75 " and depth=2.5 " | Processed casting |

M.E. - Mechanical Energy  Th. E. - Thermal Energy  E.E. - Electrical Energy

**Figure 1 – Functional model of an electric wok**




functions connected by the flows on which they operate (Stone and Wood, 2000). This structure is an easy way for a designer to see what type of functions are performed without being distracted by any particular form the artifact may take. An example functional model of an electric wok is shown in Figure 1.

*Modules*: Modules are simply clusters of functions that could be embodied by one component or assembly based on the flow(s) that the functions operate on. This is important as it suggests to the designer that a component or assembly that carries out all of the combined functions can be manufactured. Graphically, the functional model is augmented with potential modules (or partitions) based on a heuristic procedure (Stone et al., 1998; Gonzalez-Zugasti et al., 2000; Stone et al., 2000a; Stone et al., 2000b). Modules can be seen in Figure 1, denoted by boxes with hashed lines.

*Function Component Matrix*: A function-component matrix records the component(s) that solve each function. Within the matrix, rows designate product components and columns designate the sub-functions of the product. For a single component, the matrix is binary, with a "1" showing that the component solves the corresponding function and a "0" indicating no relationship. When multiple product component-function matrices are aggregated together (known as a chi-matrix) the function-component now can be used to generate concepts (Strawbridge et al., 2002). The component function matrix also serves as a roadmap linking the functional model and bill of materials. An example function-component matrix is shown in Table 4.

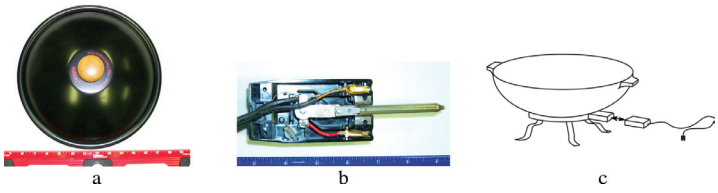
*Design Structure Matrix*: The Design Structure Matrix (DSM) is a matrix in which rows and columns represent the set of artifacts within a product (Pimmler and Eppinger, 1994). When two artifacts within a product interact with one another in some way, the cell where a row and column corresponding to those two artifacts meet is marked with a "1" (or alternatively an "X"). Cells corresponding to pairs of artifacts that do not interact are marked with a "0" (or alternatively left blank). The DSM representation results in a symmetric matrix because the interaction between artifacts A and B will show up at the intersection of row A and column B, as well as at the intersection of row B and column A. This is useful in the design process to see how artifacts within a product relate to each other physically. Table 5 shows a fragment of a DSM for an electric wok.

**Table 4 – Partial component-function matrix of an electric wok**

| FUNCTION / COMPONENT | SCREW 1 | LID HANDLE | LID | SCREWS 2 | BACK COVER FOR TEMPERATURE CONTROL | WIRE SNAPS | PLUG AND CORD | KNOB FOR TEMPERATURE CONTROL | POTENTIOMETER MECHANISM | SNAPS TO TRANSMIT EE TO THERMAL MODULE | POINTED METAL ROD TO TRANSMIT EE | LED | FRONT COVER FOR TEMPERATURE CONTROL | SCREWS 3 | METAL COVER | SCREW 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DISTRIBUTE MECH. ENERGY | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| EXPORT SOLID | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| IMPORT ELEC. ENERGY | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| IMPORT SOLID | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| INDICATE VISUAL SIGNAL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| REGULATE ELEC. ENERGY | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SECURE MECH. ENERGY | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| STABILIZE SOLID | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| STOP THERMAL ENERGY | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 5 – Partial design structure matrix of an electric wok**

| | Lid screw | Lid handle | Lid | Temp switch | Potentiometer mechanism | Plug and cord | Socket | Wire snaps | Heating coils | Metal plate | Nuts to hold base | Metal base cover | Nuts to hold metal base cover |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Lid screw | 1 | | | | | | | | | | | | |
| Lid handle | 1 | 1 | | | | | | | | | | | |
| Lid | 1 | 1 | 1 | | | | | | | | | | |
| Temp switch | | | | 1 | | | | | | | | | |
| Potentiometer mechanism | | | | 1 | 1 | | | | | | | | |
| Plug and cord | | | | | | 1 | | | | | | | |
| Socket | | | | | | 1 | 1 | | | | | | |
| Wire snaps | | | | | | | | 1 | | | | | |
| Heating coils | | | | | | | | 1 | 1 | | | | |
| Metal plate | | | | | | | | | | 1 | | | |
| Nuts to hold base | | | | | | | | | | 1 | 1 | | |
| Metal base cover | | | | | | | | | | | 1 | 1 | |
| Nuts to hold metal base cover | | | | | | | | | | | | 1 | 1 |
| Plastic supporting stand | | | | | | | | | | | | 1 | 1 |
| Handle screws | | | | | | | | | | | | | |
| Handles | | | | | | | | | | | | | |
| Threaded base metals | | | | | | | | | | 1 | 1 | 1 | 1 |
| Pan | | | | | | | | | | | | | |



a b c

**Figure 2 – Example component photos (a & b) and solid model (c) of an electric wok**

*Product Vector*: A product vector is used to determine a function's overall importance and is based on the weighted customer needs for the product. Each customer need associates with one or more functions. The weighted sub-functions are summed to give an overall importance weight for that sub-function. This is a useful step in the design stage to quickly identify key functions of a product. A product vector for an electric wok is shown in Table 6.

*Geometric Representations*: Component and artifact photos

**Table 6 – Product vector for an electric wok**

| Sub-Function/Product | Westbend Electric Wok |
|---|---|
| Distribute Mechanical Energy | 4.0 |
| Stabilize Mechanical Energy | 7.0 |
| Import Hand | 3.0 |
| Import Mechanical Energy | 4.0 |
| Import Electrical Energy | 4.0 |
| Transfer Electrical Energy | 5.0 |
| Regulate Electrical Energy | 5.0 |
| Convert Electrical Energy to Thermal Energy | 6.0 |
| Transfer Thermal Energy | 6.0 |
| Sense Thermal Energy | 4.0 |
| Import Solid | 4.0 |
| Distribute Solid | 4.0 |
| Store Solid | 8.5 |
| Export Solid | 8.5 |

along with solid models (when available) are also collected. These help the designer to visualize the form of the particular artifact/ function. Figure 2 shows an example of component photos and solid model captured for an electric wok.

## 3.2 NIST DESIGN REPOSITORY SYSTEM

All of the above identified design knowledge models exceed the representational capabilities of current commercial computational design tools. A more flexible and vendor-neutral data structure is needed to represent the heterogeneous knowledge that designers use. The NIST Design Repository Project, was initiated to meet this need (Szykman et al., 1996; Murdock et al., 1997; Szykman et al., 1999; Shooter et al., 2000; Szykman et al., 2001; Szykman, 2002).

NIST has developed a set of information models to be used for modeling product knowledge at varying levels of detail. There are several data entities which allow for a variety of aspects of a product description to be represented. The classes specified in the NIST Core Product Model include: Artifact, Function, Transfer Function, Flow, Form, Geometry, Material, Behavior, Specification, Configuration, Relationship, Requirement, Reference and Constraint[2] (Fenves, 2001). Along with these classes there is a set of specific information needed with each item and a specified type of value that can be entered.

Nearly all of the defined data types contain elements such as those shown in the Artifact Class shown in Figure 3. The Artifact class specifies that the artifact name along with artifact information and references are required. The "(I)" means that any information contained about the denoted elements is inherited from an abstract class defined separately, in this case the abstract class called DPR_Object, as indicated in the first line of the class definition. An element denoted with "# (NOT NULL)" requires that the field must contain data. The element "type" contains "[Artifact_Family]" on the same line. Instances such as this one, in which a term appears in square brackets denotes a list of such pointers; thus the "function" is defined as being a list of at least

[2] Several "abstract class" definitions also exist. These classes facilitate database design and implementation by grouping attributes common to all of the subclasses of a given class. However, instances of the abstract classes cannot be created, and are therefore not used in actual product models (Szykman et al., 2001).

```
Class Artifact      #    (inherits from DPR_Object)
    {
    name                    (I)
    information             (I)
    references              (I)
    is_referenced_by        (I)
    is_member_of            (I)
    is_special_member_of    (I)
    type                    [Artifact_Family]       #    (NOT NULL)
    is_specified_by         {[Specification]}
    config_info             [Config_Info]           #    (NOT NULL)
    function                {[Function]}            #    (NOT NULL)
    form                    [Form]                  #    (NOT NULL)
    behavior                {[Behavior]}
    subartifacts            {[Artifact]}
    subartifact_of          {[Artifact]}
    is_source_of            {[Flow]}
    is_destination_of       {[Flow]}
    }
```

**Figure 3 – Artifact class definition**

```
<DesignRepository>
    <Artifacts>
        <Artifact name="widget" type="Artifact>
            <Information>
                <Description>
                    description of widget
                </Description>
                <Creation>
                    <ref:Person ref="name_who_entered"/>
                    <Date value="Mon Jan 1 10:10:10 EST 2000"/>
                </Creation>
                <LastUpdate>
                    <ref:Person ref="name_who_entered"/>
                    <Date value="Mon Jan 1 10:10:10 EST 2000"/>
                </LastUpdate>
            </Information>
            <ref:Function ref="widget_1_Function"/>
            <ref:Form ref="widget_1_Form"/>
            <ref:Behavior ref="widget_1_Behavior"/>
            <subartifacts>
                <ref:Artifact ref="widget_part_1"/>
            </subartifacts>
        </Artifact>
```

**Figure 4 – Sample NIST formatted XML data**

one (because of the "NOT NULL") or more pointers to items belonging to the "Function" class.

The NIST design repository representation model is a basic framework to help guide what type of product information is collected and how the elements of information are related to each other. NIST has also developed a mapping from this representational framework into an XML data format. Within the NIST XML code there are five different sections: Artifacts, Functions, Forms, Behaviors and Flows. These sections contain information relative to their denoted naming system.

Figure 4 shows a sample of NIST-formatted XML. The opening tag is "<DesignRepository>" followed by "<Artifacts>" and "<Artifact name=…>." These first few tags begin the XML representation of the "DesignRepository" and then define the "Artifacts" subset followed by a specific "Artifact" named "widget." Within the "widget" artifact there are tags for information fields that contain a description of the artifact, the creation date of the data entity, who created the entity when it was last updated, and who performed the update. The "<ref:…….>" tags are generic reference tags within the XML language, used in this context to link one data entity (e.g., the "widget" artifact) to other entities that are associated with it but are defined elsewhere (e.g., the widget's corresponding function, form and behavior). These entities all have information in their appropriate section of the XML file. Subartifacts of the "widget" are contained within the "<subartifacts>" tagged area and are then referenced by the actual

artifact. The structure of XML is such that an unlimited number of unique "Artifact" elements can be defined under the <Artifacts> tag. Every element keyed before "<Artifacts/>," the close of the artifacts tag is described underneath the "Artifacts" umbrella. This structural style also follows for "Functions" contained underneath the "<Functions>" tag, and likewise for Forms, Behaviors and Flows. A single product represented in this XML format will span thousands of lines of data to accurately represent the product.

### 3.3 TECHNOLOGY

XML (eXtensible Markup Language) is a basic markup language for documents containing structured information (W3C, 2000). Structured information includes text, graphics and other elements and the structure comes from the linking that gives an indication of what role the data plays. This information comes from the tags contained within the XML format. XML is similar to HTML in that it uses tags to "markup" elements. A markup language is a mechanism to identify structures in a document. XML differs from HTML by virtue of its extensibility. Unlike HTML, which has a fixed set of tags, XML allows for the creation of user-defined tags. The XML specification simply defines a standardized method to add markup to documents.

XSL (Extensible Stylesheet Language) is a language for expressing stylesheets given a specified class of arbitrarily structured XML data (W3C, 2001). XSL is used in conjunction with an XSL processor and XML data. An XSL stylesheet processor accepts a document or data in XML, along with an XSL stylesheet and produces an output that is formatted as defined by the stylesheet. A stylesheet is essentially a document roadmap, it defines what elements are, where they can be found and where they need to go. An XML file can be parsed with an XSL stylesheet and passed through a stylesheet processor to create reformatted XML, basic text, HTML or graphics. XSLT (XSL Transformations) provide a language that is used to define transformations of one XML file into another (W3C, 2002). XSLT is an important part of XSL.

## 4.0 RESEARCH METHOD

This section begins with a summary of the practice of design data entry and usage in the knowledge-based system previously developed at UMR. The shortcomings of the previous system are summarized in section 4.1. Section 4.2 presents a framework for an enhanced design repository-based system with ease of entry and design tool output capability. Finally the solution is implemented for an improved repository entry, management and retrieval method. The system architecture is described in section 3.3 and details of its underpinnings and output capabilities are described in sections 4.4-5.

### 4.1 OBSERVATION

Approaching this as a design problem, a list of customer needs for an effective design repository is initially developed. The most important customer need is to improve the repository data entry method. In order to improve data entry, researchers at UMR reviewed the current design tools and determined what pieces of information are required to fully represent product knowledge. The data sets recorded contain customer needs lists, bills of materials, functional models, module based functional models, function-component matrices, design structure matrices, product vectors, artifact photos and assembly instructions. Although this information is valuable to the design repository, it is sometimes redundant in explanation and creation. Most of these elements are currently created inside of spreadsheet and drawing applications. Some application dependence is removed by exporting the final files to an Adobe PDF (Portable Document Format); however manipulation of the design knowledge still requires the original applications. Product of design tools from the repository is often a tedious process. For example, an aggregate component-function matrix was populated manually by dragging and dropping individual product component-function matrices into the combined component-function matrix.

The goal of the research described in this paper is to reduce workload of design engineers to populate design repositories, and, thus make them more appealing as a design tool. It was found that the bill of materials and a functional model are the key pieces of information that are required to generate a majority of the remaining representation schemata.

The most noticeable issues with the previously existing data were the inconsistency of product representation language and format. This design repository effort at UMR has been in existence since the summer of 2000 and had over 11 different researchers contributing product data. The inconsistency between artifact representations can be linked to the time span and number of researchers associated with the repository. This indicated that a strict framework was needed in order to unify the repository format and increase consistency.

Physical artifact information such as dimensions, material and the manufacturing process varied greatly. Most artifact dimensions were represented by "L=6, W=2, H=3." Noticeably the lack of associated units of measure results in an imprecisely specified product description. Multiple artifact dimensions were often keyed into a single cell rather than individual cells, with no consistent use of labeling. Similar issues were present with material and manufacturing process data. Also noticed across the repository was a great discrepancy between use of various function levels (i.e. primary, secondary or tertiary) for modeling. According to Hirtz et al. (2000) the secondary level of function is the preferred level for artifact representation. Another issue encountered was inconsistencies in the use of abbreviations for functions, such as "Ex" for Export, "Im" for Import, "Sep" for Separate, or "Dist" for Distribute.

The textual information within the bill of materials combined the input flow along with the function into a single cell. An example of this is "Conv. EE to ME" which represents "Convert Electrical Energy to Mechanical Energy." Often an artifact has multiple sub-functions and input flows, which took a similar representation form as "Conv. EE to ME" combined together and sometimes separated by commas or colons. The data types and extensive variations in representation would not allow for even an intelligent parser to retrieve accurate information.

Another hurdle to tackle was deciding how much textual product information would be necessary to automatically generate a functional model, a function component matrix and other representation types described in section 3. The functional model of a given product includes subfunctions as well as input and

output flows. In order to represent this type of information digitally, both input and output flow must be recorded, along with the given artifact's subfunction. These artifacts and associated subfunctions generally have input and output flows linking them to other artifacts and subfunctions, but can also interface with an environment outside of the functional model (import/export). In the repository's initial state, this type of artifact linking had not been captured.

From the initial examination of product data and review of current design representation schemas, an extensive list of areas for improvement was identified. The next section presents a formulation of an improved repository knowledge entry scheme to address these issues.

## 4.2 HYPOTHESIS (SOLUTION FORMULATION)

The approach taken by researchers at UMR to address this problem was the development of a single application called the Enhanced Bill of Materials (EBOM), which handles entry, management and export of repository knowledge. This approach somewhat parallels the original repository editor of the NIST Design Repository Project (Szykman, 2002). All design knowledge corresponds to a single artifact (which may itself be composed of additional subartifacts), suggesting that an efficient EBOM should similarly be focused around each single artifact. To accurately represent design knowledge and to embody flow origin and destination information, an extensive representation for each artifact must be created. The layout must include the standard elements of a BOM, along with additional information regarding the flow paths so that an accurate digital representation of the product can be achieved.

A test bed of ten products was chosen to create the initial implementation of the EBOM. These ten products were chosen because they contained artifact representations spanning nearly all classes of flows and subfunctions and they were generally electromechanical consumer products. Having a test data-set that maps into each of the function and flow types ensured that the most possibilities and combinations of artifact setup were taken into consideration while developing a new repository entry method.

The ten products were used both to test the integrity of the data, and for a lesson in database programming. The test products' Microsoft Excel spreadsheets were aggregated into a large spreadsheet and then imported into a FileMaker Pro database. Attempts at exporting XML, Excel and HTML data were conducted within FileMaker Pro (from FileMaker Inc.). Importing the data into a formal database and the simple export tests demonstrated the strengths and weaknesses of the database and the information contained within.

**Table 7 – Material type master list**

| Material |
|---|
| Plastic |
| ABS |
| Nylon |
| Aluminum |
| Steel |
| Rubber |
| Wood |
| Concrete |
| Composite |
| Metal |
| Foam |
| Glass |
| Iron |

**Table 8 - Manufacturing processes master list**

| Manufacturing Process |
|---|
| Injection_molding |
| Stamping |
| Extrusion |
| Rolling |
| Casting |
| Forging |
| Machining |
| Forming |
| OEM |

To eliminate discrepancies in language representation, abbreviations and formatting, defined lists of commonly used BOM elements along with a structured set of the functional language terms must be implemented. This is achieved through the use of master lists within a relational database. This ensures that all products and artifacts represented across the entire span of repository data are consistent, not only in language but also in format. For example, the material and manufacturing process fields are limited to a pre-defined list of commonly used materials and manufacturing techniques. Because these lists are maintained in separate relational databases, whenever the master list is updated it will propagate to all the products. The initial material master list shown in Table 7, was created by examining the most commonly used materials that were already represented within the repository. The manufacturing process list shown in Table 8, was adapted from the Dixon and Poli (1995) Taxonomy of Manufacturing, and was combined with commonly seen processes from previous product dissection. Master lists were also created for flow and function secondary classes using the Hirtz et al. (2002) functional basis. In addition, these lists can easily be edited to expand the terms available to users.

## 4.3 SYSTEM ARCHITECTURE

To implement the EBOM formulation, a database was created using FileMaker Pro. The main user interface screen, shown in Figure 5, allows product knowledge to be entered on an artifact-by-artifact basis. The *Artifact Name* field is a simple text field used to type a common name for the represented artifact. The *Part Number* field is a simple indexing serial number to track part changes. Although this number is indexed, it can be manually overridden by entering a different numeric value. The *Sub Artifact Of* field is where a higher-level artifact can be associated with the current artifact. This list can be typed in manually, however, the field has a drop/select menu associated with it. The drop/select menu pulls textual data from all of the artifacts already entered for the corresponding product. If an assembly to sub-assembly to component sequence is followed when entering a product the *Sub Artifact Of* field along with the *Part Number* field will automatically allow for the current artifact's predecessor to be easily selected and the part number will order correctly. This automated numbering and name selection feature greatly decreases the product knowledge entry time.

The *Input Artifact, Input Flow, Sub Function, Output Flow* and *Output Artifact* fields are used to trace product flow through their corresponding artifacts and subfunctions. On a typical BOM only the input flow and subfunction of an artifact are recorded and usually represented like "input solid." This type of information is descriptive, however there is not enough information to create a string of artifacts and functions within a given product. The EBOM method overcomes this limitation. As described above, the need for consistent data format for the physical parameters is a must. Here the physical parameters are a more abstract representation of the artifact, often capturing the dimensions of the

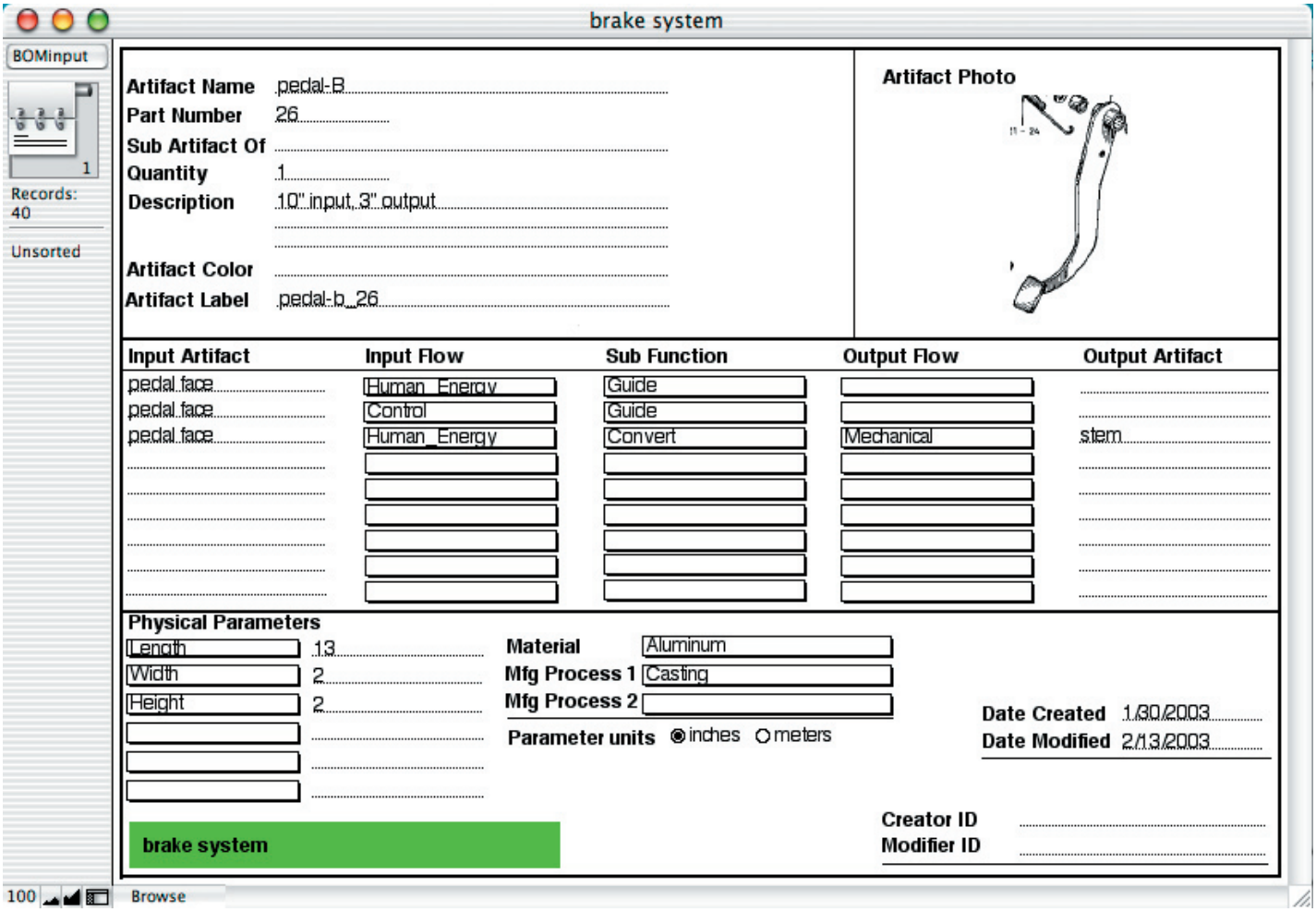



**Figure 5 – Repository artifact input screen**

bounding volume and key feature dimensions. A more detailed CAD/Solid model can be associated with the artifact as well by attaching the CAD/Solid model using the "Browse" button. To unify the parameter description, auto-select fields have been added. The user simply chooses what type of measurement is being recorded from a pre-defined list of dimensioning variables. After selecting the type of dimension variable, the user then inputs the dimensions corresponding numeric value. The database can hold up to five unique dimensions per artifact, allowing for almost any type of product to be dimensioned properly.

Referring again to Figure 5, the middle section of the screenshot of the main product entry page shows how functional representations of the artifact are entered. The EBOM has placeholders for the *Input Artifact* and its *Input Flow* as well as the *Sub Function* of the artifact being described with its *Output Flow* and *Output Artifact* fields. After information about the destination/origin artifacts and flows and the subfunction of an artifact have been entered, enough information exists to create the function-flow referencing key, a unique key that attaches the subfunction description to a specific artifact.

## 4.4 UNDER THE HOOD

The function-flow referencing key uses its naming structure from the *Artifact Label* field, which is automatically created upon entering a product name and cannot be altered by the user. The *Artifact Label* that is created removes any abbreviating characters, replaces spaces with underscores and removes all symbols from the *Artifact Name* field. The *Artifact Label* also attaches an underscore and a numeric value at the end of the modified artifact name, corresponding to the artifact's record number within the product's database file. It is this key that is used to create the flow-function-flow string. The string elements are separated by commas so that they can be parsed out for further manipulation.

An example of the referencing string is given below:
flow_"Input Flow"+_name_+"Input Artifact Label", sub_"Sub Function"+_name_+"Artifact Label", flowout_"Output Flow"+_name_+"Output Artifact Label"

Although the user sees the plain English name when selecting or keying elements into the input or output artifact fields they are actually telling the current record which *Artifact Label* name to include in the flow-function-flow keys. This information is not particularly useful while the data is within FileMaker because there is already inherent referencing within the database structure. The function-flow-function output is useful when data from FileMaker is exported as a general text, comma/tab delimited or XML file. At this point the string is then useful to text/XML parsers to reorganize, structure and further format the data.

FileMaker also allows for an artifact photo to be attached

alongside the corresponding artifact data, this is useful when exporting because the photo can be exported to a media folder renamed with the *Artifact Label* field with a .jpg extension. This allows an external browser to view images by being linked based on name commonality. The *Creator ID* and *Modifier ID* fields can be automatically entered if an option is enabled to make the user login to the database with name and password. The *Date Created* and *Date Modified* fields are automatically generated values that are created or updated when original or repeated record activity takes place.

Currently each individual product is maintained in a separate database file. As the number of products increases, it becomes a massive undertaking to update or change all of the individual database files. This is overcome by using AppleScript (by Apple Computer), a system and application scripting language, to automatically make changes to all of the individual database files.

## 4.5 SYSTEM OUTPUT

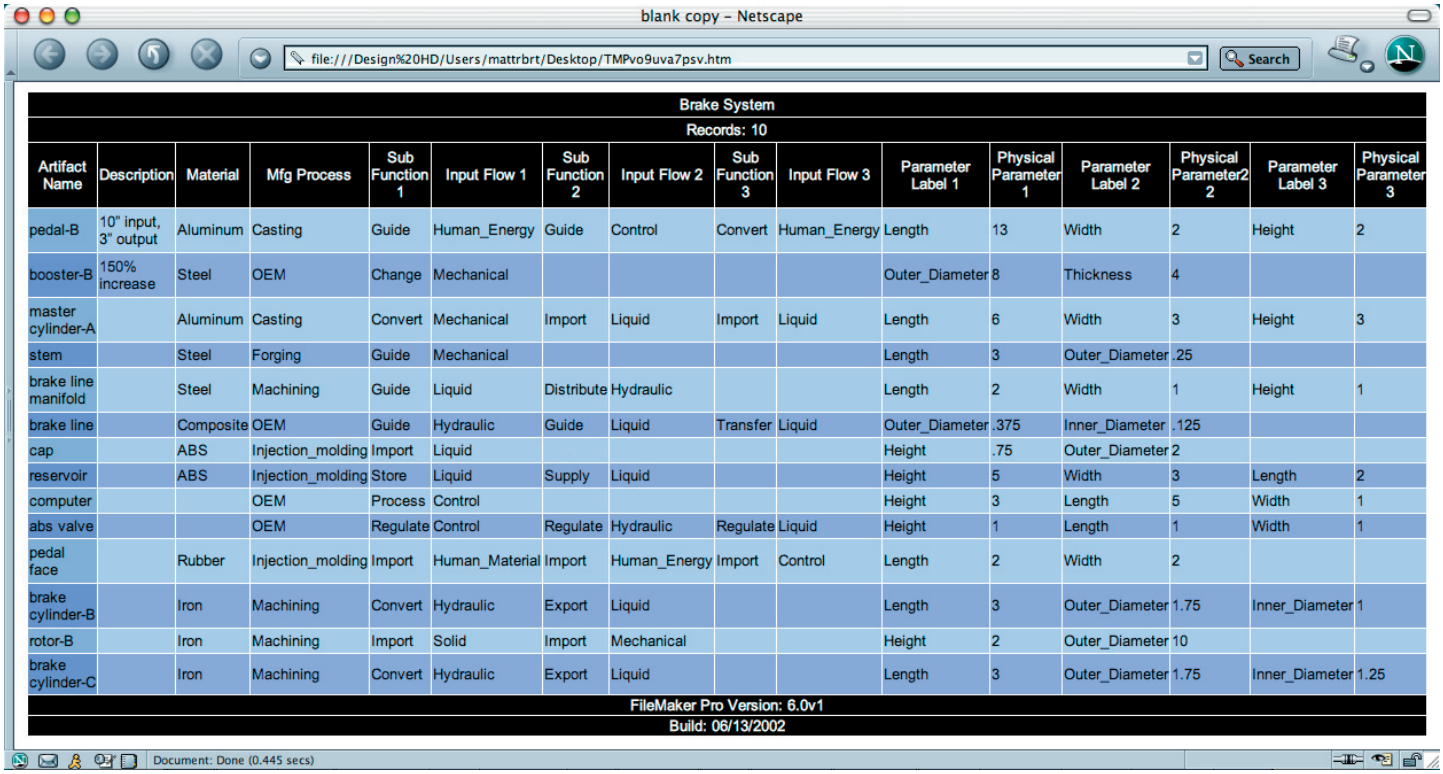
This database program is an excellent gateway for original artifact information input, or to import and edit information from existing file structures. FileMaker does not, however, allow for easy manipulation of variables, customized detailed searches or data-dependent output display. All of the information that is entered into FileMaker is transformed into a neutral Unicode Text basis so that it can be parsed and easily read by other file formats and programs. The main intent of this form of entry is to generate XML or other platform-independent data. Although the user may see artifacts represented in plain English format, the Unicode text naming structure behind FileMaker is necessary to create a platform-independent data set.

Following NIST's approach toward neutral data exchange, XML was identified as the desired output format from the FileMaker-based design repository. Other common exports from FileMaker are usually in the form of comma or tab delimited files that can easily be imported to Microsoft Excel.

When choosing to export XML data from FileMaker, it is necessary to specify an XSL schema to follow. FileMaker has the capability to export its own semi-proprietary XML data format. The XML that FileMaker exports is syntactically valid, but FileMaker does not easily recognize the database's field naming structure without an XSL schema to parse the output into a commonly recognizable form of XML. Having data in an XML format is a very powerful and effective tool for moving the data into other forms and transporting to various database systems.

XSLT is necessary to do any of these XML data transformations. XSL is a file mapping roadmap between the input format and the desired output format. An XSL file combined with an XML file can be passed through an XML parser. The parser will check that the XML and XSL files are in a valid format and match each other. The latest version of the XML standard is version 2.0, which allows outputs ranging from a general text format to a scalable vector image. Commonly-used parsing engines include Sun Microsystems' Xerces and Xalan engines. There are many other proprietary parsers that generally adhere to XML but neglect some of the strict rules imposed by XML 2.0.

By outputting all of the FileMaker product databases into XML they can all be merged into a single XML data file. The data will not overlap or overwrite other data because each artifact and its associated information will be tagged within an individual product opening and closing tag. Having the multiple product repositories represented in a single XML file is useful when it is necessary to search across the entire set of artifacts for particular




Brake System
Records: 10

| Artifact Name | Description | Material | Mfg Process | Sub Function 1 | Input Flow 1 | Sub Function 2 | Input Flow 2 | Sub Function 3 | Input Flow 3 | Parameter Label 1 | Physical Parameter 1 | Parameter Label 2 | Physical Parameter2 2 | Parameter Label 3 | Physical Parameter 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| pedal-B | 10" input, 3" output | Aluminum | Casting | Guide | Human_Energy | Guide | Control | Convert | Human_Energy | Length | 13 | Width | 2 | Height | 2 |
| booster-B | 150% increase | Steel | OEM | Change | Mechanical | | | | | Outer_Diameter | 8 | Thickness | 4 | | |
| master cylinder-A | | Aluminum | Casting | Convert | Mechanical | Import | Liquid | Import | Liquid | Length | 6 | Width | 3 | Height | 3 |
| stem | | Steel | Forging | Guide | Mechanical | | | | | Length | 3 | Outer_Diameter | .25 | | |
| brake line manifold | | Steel | Machining | Guide | Liquid | Distribute | Hydraulic | | | Length | 2 | Width | 1 | Height | 1 |
| brake line | | Composite | OEM | Guide | Hydraulic | Guide | Liquid | Transfer | Liquid | Outer_Diameter | .375 | Inner_Diameter | .125 | | |
| cap | | ABS | Injection_molding | Import | Liquid | | | | | Height | .75 | Outer_Diameter | 2 | | |
| reservoir | | ABS | Injection_molding | Store | Liquid | Supply | Liquid | | | Height | 5 | Width | 3 | Length | 2 |
| computer | | | OEM | Process | Control | | | | | Height | 3 | Length | 5 | Width | 1 |
| abs valve | | | OEM | Regulate | Control | Regulate | Hydraulic | Regulate | Liquid | Height | 1 | Length | 1 | Width | 1 |
| pedal face | | Rubber | Injection_molding | Import | Human_Material | Import | Human_Energy | Import | Control | Length | 2 | Width | 2 | | |
| brake cylinder-B | | Iron | Machining | Convert | Hydraulic | Export | Liquid | | | Length | 3 | Outer_Diameter | 1.75 | Inner_Diameter | 1 |
| rotor-B | | Iron | Machining | Import | Solid | Import | Mechanical | | | Height | 2 | Outer_Diameter | 10 | | |
| brake cylinder-C | | Iron | Machining | Convert | Hydraulic | Export | Liquid | | | Length | 3 | Outer_Diameter | 1.75 | Inner_Diameter | 1.25 |

FileMaker Pro Version: 6.0v1
Build: 06/13/2002

**Figure 6 – HTML table output of a BOM**

```
<Artifacts>
    <Artifact name="dirt_cup1" type="Artifact">
        <Information>
            <Description>
            holds debris
            </Description>
            <Creation>
            <ref:Person ref="UMR Design Repository"/>
            <Date value="3/7/2002"/>
            </Creation>
            <LastUpdate>
            <ref:Person ref="Matt Bohm"/>
            <Date value="3/7/2002"/>
            </LastUpdate>
        </Information>
        <ref:Function ref="store_debris1"/>
        <subartifacts>
            <ref:Artifact ref="dirt_cup1">
        </subartifacts>
    </Artifact>
    <Artifact name="check_valve_clips" type="Artifact">
        <Information>
```

**Figure 7 – Portion of an artifact in NIST XML representation**

functions, or to create an extensive function-component matrix. The repository information in a single XML file can also be passed to a SQL (MySQL), WebObjects (Apple Computer) or another database server. From the server a wide variety of repository sorting and outputs can be performed. By creating JSP (JavaServer Pages) the XML from the database server can be viewed or sorted and then passed through XSLT stylesheets and viewed as HTML through a standard web browser (Sun Microsystems, 2003).

With the implementation approach described above, product information can be easily exported into a variety of formats ranging from a brief product overview to a thorough product layout. An example of one such export into a bill of materials is shown in Figure 6. This bill of materials was created by exporting XML data and then performing an XSL transformation which converted the data into an HTML table.

Product information from a database can also be exported into the XML-based format developed as part of the NIST Design Repository Project using the appropriate XSL transform sheet. Figure 7 shows a portion of an artifact represented in the XML version of NIST's design repository data format.

## 5 CONCLUSIONS

The implementation of this project has significantly increased the usefulness of repository of design knowledge developed at UMR in several ways. The first and most noticeable aspect are the types of product design knowledge and the way in which a designer records this knowledge. The workload of entering product knowledge into a design repository is greatly reduced. The previous approach required seven separate data entry sets over a variety of applications. The EBOM entry system reduces that to one. Because the repository can export multiple types of data and data formats, the need to manually create individual bill of materials, design structure matrices and component function matrices is eliminated. The repository is now capable of importing and exporting product knowledge in the NIST design repository format. A simpler approach to product dissection also results. The repository information entry scheme allows a researcher to focus on a single artifact at a time and automatically keeps track of component interactions, providing a big picture view of the product once all product artifacts have been entered. Overall, the repository system creates a unified point of access with which to interact and control all of the contained data.

Currently the process of combining multiple repositories into a single XML file for the purpose of web-based access is indirect, and must be performed manually when product files are updated or created. Although many steps have been taken to ensure that artifact referencing is unique, there can exist rare circumstances where referenced artifacts may conflict when joined together with other design repositories. A fully integrated repository manager must be created to check across the entire database of products to validate and prompt the user if duplicate product referencing exists to eliminate this potential hazard. A repository database manager could also complete such tasks as automatically updating web content when repository knowledge is added or altered.

Looking at the big picture, several important types of design knowledge used by designers have been determined, and redundant information in the representations has been identified and distilled down to one core set of required knowledge. This greatly reduces the repository data entry workload for designers. From this knowledge set, several key types of representations and design aids (discussed earlier in section 2.3) can be generated and used.

## 6 FUTURE WORK

Future work includes increasing the number of design tools that can be directly exported from the repository, as well as integrating more knowledge representations into it. In particular, the EBOM entry system is currently being expanded in order to be able to accept performance equations for each artifact. Once implemented, models of system performance can be constructed for components or, more abstractly, for a chain of functions. Also, component failure data will be integrated into the repository structure to support failure analysis techniques with actual occurrence data.

The repository project thus far has been based on a target requirement of containing only a few hundred unique products. Ultimately the intent is to create a design repository system capable of managing thousands of products. To handle this much larger system a standalone Java application is envisioned, which will be capable of storing and retrieving data to and from an SQL database server. The Java application would be platform-independent, and capable of operating on numerous clients accessing a single database of information.

## REFERENCES


Campbell, M., Cagan, J. and Kotovsky, K. (2000), "Agent-based Synthesis of Electro-Mechanical Design Configurations," *Journal of Mechanical Design*, 122(1): 61-69.

Campbell, M., Cagan, J. and Kotovsky, K. (2001). "Learning From Design Experience: Todo/Taboo Guidance," *Proceedings of the 2001 Proceedings of the 2001 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, DETC01/DTM-21687, Pittsburgh, PA.

Dixon, J. and Poli, C. (1995), *Engineering Design and Design*

*for Manufacturing: A Structured Approach*, Conway, MA, Field Stone.

Fenves, S. J. (2001), A Core Product Model for Representing Design Information, NISTIR 6736, National Institute of Standards and Technology, Gaithersburg, MD, April

Gonzalez-Zugasti, J. P., Otto, K. N. and Baker, J. D. (2000), "A Method for Architecting Product Platforms," *Research in Engineering Design*, 12(2): 61-72.

Hauser, J. and Clausing, D. (1988), "The House of Quality," *Harvard Business Review*, 66(3): 63-73.

Hirtz, J., Stone, R., McAdams, D., Szykman, S. and Wood, K. (2002), "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," *Research in Engineering Design*, 13(2): 65-82.

McAdams, D. and Wood, K. (2000). "Quantitative Measures For Design By Analogy," *Proceedings of the 2000 Proceedings of DETC2000*, DETC2000/DTM-14562, Balitmore, MD.

Murdock, J., Szykman, S. and Sriram, R. (1997). "An Information Modeling Framework to Support Design Databases and Repositories," *Proceedings of the 1997 Proceedings of DETC'97*, DETC97/DFM-4373, Sacramento, CA.

NIST (2000). *Workshop on Product Representation for Next-Generation Distributed Product Development*. Gaithersburg, MD, National Institute of Standards and Technology.

Otto, K. (1996). "Forming Product Design Specifications," *Proceedings of the 1996 Proceedings of the 1996 ASME Design Theory and Methodology Conference*, Irvine, CA.

Otto, K. and Wood, K. (2001), *Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development*, New York, Prentice-Hall.

Pahl, G. and Beitz, W. (1996), *Engineering Design: A Systematic Approach*, Springer Verlag.

Pimmler, T. and Eppinger, S. (1994). "Integration Analysis of Product Decompositions," *Proceedings of the 1994 Proceedings of the ASME Design Theory and Methodology Conference*, DE-Vol. 68.

Sheppard, S. D. (1992). "Mechanical Dissection: An experience in how things work,," *Proceedings of the 1992 Proceedings of the Engineering Education: Curriculum Innovation & Integration*, Santa Barbara, CA.

Shooter, S., Keirouz, W., Szykman, S. and Fenves, S. (2000). "A Model For Information Flow In Design," *Proceedings of the 2000 Proceedings of the ASME Design Theory and Methodology Conference*, DETC2000/DTM-14550, Baltimore, MD.

Stock, M., Stone, R. and Tumer, I. Y. (2003). "Going Back in Time to Improve Design: The Function-Failure Design Method," Submitted to *Proceedings of the 2003 ASME Design Engineering Technical Conference, Design Theory and Methodology Conference*, Chicago, IL.

Stone, R. and Wood, K. (2000), " Development of a Functional Basis for Design," *Journal of Mechanical Design*, 122(4): 359-370.

Stone, R., Wood, K. and Crawford, R. (1998). "A Heuristic Method to Identify Modules from a Functional Description of a Product," *Proceedings of the 1998 Proceedings of DETC98*, DETC98/DTM-5642, Atlanta, GA.

Stone, R., Wood, K. and Crawford, R. (2000a), "A Heuristic Method for Identifying Modules for Product Architectures," *Design Studies*, 21(1): 5-31.

Stone, R. B., Wood, K. L. and Crawford, R. H. (2000b), "Using quantitative functional models to develop product architectures," *Design Studies*, 21(3): 239-260.

Strawbridge, Z., McAdams, D. A. and Stone, R. B. (2002). "A Computational Approach to Conceptual Design," *Proceedings of the 2002 ASME Design Engineering Technical Conference, Design Theory and Methodology Conference*, DETC02/DTM-34001, Montreal, Canada.

Sun Microsystems, Inc. (2003). *The JavaServer Pages Specification, Version 2.0*, http://jcp.org/en/jsr/detail?id=152.

Szykman, S. (2002). "Architecture and Implementation of a Design Repository System," *Proceedings of the 2002 Proceedings of DETC2002*, DETC2002/CIE-34463, Montreal, Canada.

Szykman, S., Fenves, S., Keirouz, W. and Shooter, S. (2001), "A Foundation for Interoperability in Next-Generation Product Development Systems," *Journal of Computer Aided Design*, 33: 545-559.

Szykman, S., Racz, J. and Sriram, R. (1999). "The Representation of Function in Computer-Based Design," *Proceedings of the 1999 Proceedings of the ASME Design Theory and Methodology Conference*, DETC99/DTM-8742, Las Vegas, NV.

Szykman, S., Sriram, R. and Smith, S. (1996). "Proceedings of the NIST Design Repository Workshop," *Proceedings of the 1996 Gaithersburg, MD*.

Urban, G. and Hauser, J. ( 1993), *Design and Marketing of New Products*, New York, Prentice Hall.

W3C (2000), *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, World Wide Web Consortium, http://www.w3.org/TR/REC-xml.

W3C (2001), *Extensible Stylesheet Language (XSL) Version 1.0*, W3C Recommendation, World Wide Web Consortium, http://www.w3.org/TR/xsl.

W3C (2002), *XSL Transformations (XSLT) Version 2.0,* W3C Working Draft, World Wide Web Consortium, http://www.w3.org/TR/sxlt20/.

Wood, W. H., Gietka, P. and Verma, M. (2002). "Functional Modeling, Reverse Engineering, and Design Reuse," *Proceedings of the 2002 ASME Design Engineering Technical Conference, DTM*, Montreal, Canada.

Wood, W. H. and Verma, M. (2000). "A Function-Based Approach To Design For Manufacturing," *Proceedings of the 2000 ASME Design Engineering Technical Conference, DFM*, Baltimore, Maryland.